kpeaks: An R Package for Quick Selection of K for Cluster Analysis

Zeynel Cebeci
Div. of Biometry and Genetics
Çukurova University
Adana, Turkey
zcebeci@cukurova.edu.tr

Cagatay Cebeci
Dept. of Electronic & Electrical Eng.
University of Strathclyde
Glasgow, UK
cagatay.cebeci@strath.ac.uk

***Abstract*— The argument *k* is a mandatory user-specified input argument for the number of clusters which is required to start all of the partitioning clustering algorithms. In unsupervised learning applications, an optimal value of this argument is generally determined by using any of the internal validity indexes. However, the determination of *k* with aid of these indexes are computationally very expensive because they compute a *k* value using the results after several runs of a clustering algorithm. On the contrary, the package 'kpeaks' enables to estimate *k* before starting a clustering session. It is based on a simple novel technique using the descriptive statistics of peak counts of the features in datasets. In this paper, we introduce and illustrate the details of R package 'kpeaks' as an implementation for quick selection of the number of clusters for starting cluster algorithms.**

***Index Terms*— number of clusters, cluster analysis, partitioning clustering, unsupervised learning.**

I. Introduction

Clustering is highly demanded in data mining applications. The determination of *k*, the optimal number of clusters, in a dataset is one of the main issues in cluster analysis. This number is needed by the partitioning clustering algorithms such as K-means, Fuzzy C-means, Possibilistic C-means and their various variants. The argument *k* must be assigned as one of the mandatory input arguments to start the partitioning algorithms. Although it should be known before starting a cluster analysis, there is no definitive solutions for determining it.

The partitioning algorithms produce a clustering result either *k* be accurately or inaccurately selected. Thus, the selection of an appropriate *k* value equal to or near to the actual number of clusters is a crucial step in a clustering task before running the cluster algorithm because the quality of clustering depends on the optimal choice of this input argument. So, the number of clusters in a dataset should be determined or estimated for achieving high quality clustering results. Various subjective and objective methods can be used to determine the value of *k*. In general, the subjective methods are based on heuristic approaches to explore the underlying structure of datasets by means of various exploratory graphics [1]. In this case, some degree of experience and domain knowledge might come in handy. The subjective methods may result with poor quality clustering since the clustering algorithms may produce different results depending on the shapes and orientations of the clusters in datasets [2].

The objective methods are mainly based on cluster validation tasks by applying the validity indices after the clustering result is obtained. These indices can be classified into three groups as the external, internal and relative indices [3], [4]. The external indices use some kind of external information associated with data instances. They compare the cluster labels found by a clustering analysis to the already known class labels, which can be used as the external information for choosing an appropriate *k* value. In practice, since the external information is often not available with data, the internal validity indices become the only applicable options to reveal the quality of the clustering by using the results obtained directly from datasets themselves [5]. Finally, the relative indices are the validity measures based on comparisons of clustering results by running one or more clustering algorithms with different input parameters on the same dataset. For instance, the best partitioning is determined by comparing the objective function values which are calculated in multiple runs of a clustering algorithm.

Cluster analysis is an unsupervised learning task in which clustering tendency is previously unknown. Therefore, most of the studies focus on the internal validity indices to validate the clustering results. These indices are generally based on the compactness, separation and their combinations. Compactness is a measure of how closely related or coherent the instances are each other. Separation, on the other hand, is a measure of how the clusters are separated from each other. There are lots of internal and external validity indices introduced in the literature [4], [6], [7].

Although the internal indexes are often used to determine *k*, they are time consuming because they demand to run the clustering algorithms for several times with different values of *k* set by the users. Consequently, using the internal indexes becomes computationally ineffective to determine the *k* value in multidimensional datasets. Thus, the methods giving an appropriate number of clusters before clustering may contribute much to make clustering less time consuming. In this paper,

978-1-5386-6878-8/18/$31.00 ©2018 IEEE

'kpeaks' which is an R [8] package enabling fast and accurate estimation of k for partitioning cluster analyses has been introduced. The techniques implemented in the package are based on the use of descriptive statistics of the peaks count of the features in multidimensional large datasets.

II. Proposed Technique to Find Number of Clusters

The proposed technique for estimation of k is based on various descriptive statistics of the peak counts in the frequency distributions of the features in a dataset. As it is known very-well, the peaks in a frequency distribution indicate the modality of this distribution. For a normally distributed feature there is only one peak in its frequency distribution. Unlike a unimodal normal distribution if frequency distribution of a feature in the examined dataset is multimodal it is considered as a mixture Gaussian distributed feature consisting of different number of subgroups or clusters. Using this idea, when the peaks in a frequency distribution is found and counted, this information could be used in order to obtain the estimates of k representing the number of actual clusters in datasets.

The 'kpeaks' is an R package with a set of functions implementing the approach described above. The principal function of the package is the function called `findk` computing various estimates of k. In the package, for a given dataset, the frequency distributions of the features in the examined dataset are created firstly. The peaks are counted for each feature and stored in a peaks count vector. Then, the descriptive statistics are computed by using the peaks count vector. In the equations in Table 1, p, f_i and $f_{(i)}$ stand for the number of features, the unordered and ordered frequency of peaks of i^{th} feature, respectively. The short names of the descriptive statistics in the first column of Table 1 are used as the names of outputs of the function `findk` of 'kpeaks'. Finally, the estimates of k which are listed in Table 1 are returned as the output.

TABLE I. Options to Estimate of K with Kpeaks

Options	Description	Formula
AM	Arithmetic mean of peak counts	$1/p\ (\sum_{i=1}^{p} f_i)$
MPPC	Overall mean of the means of peak pairs	$1/\left(\frac{p^2-p}{2}\right)\left(\sum_{i=1}^{p-1}\sum_{j=i+1}^{p}(f_i+f_j)/2\right)$
MED	Median of peak counts	$f_{(\frac{n+1}{2})}$ or $1/2\left(f_{(\frac{n}{2})}+f_{(\frac{n}{2}+1)}\right)$
MOD	Mode of peak counts	f_{mod}
CIQR	Centre of the IQR of peak counts	$1/2(Q3_f - Q1_f)$
CR	Mean of the extremes of peak counts	$1/2(f_{min} + f_{max})$
MQ3M	Mean of the Q3 and maximum peak count	$1/2(Q3_f + f_{max})$
MTL	Mean of the two biggest peak counts	$1/2(f_{(n-1)} + f_{(n)})$

III. Package Functionality and Illustrative Examples

The package 'kpeaks' contains six functions and one synthetically created dataset for testing purposes. The names and functionalities of the package components are listed in Table 2. As it is seen in Fig. 1, the function `findk` returns an estimation list of k including the proposed ones for a given dataset. With the 'kpeaks' the first step for k selection is to generate the classes of each feature in an examined dataset. This process is achieved by calling the function `genpolygon`. The frequency distributions of the features produced by `genpolygon` is passed to the function `findpolypeaks`. Then, the peaks in frequency distributions of the features are found and counted by with the function `findpolypeaks`.

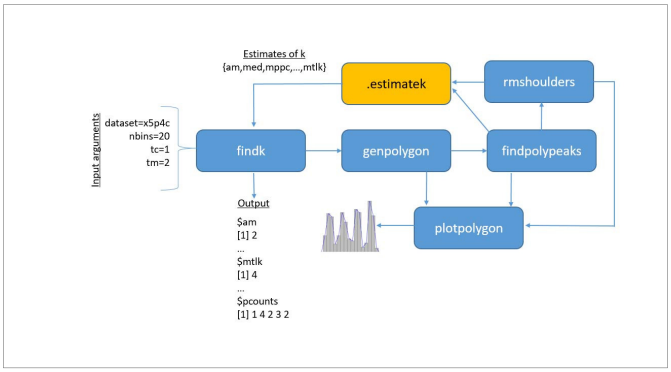

Fig. 1. Block diagram of kpeaks framework

If necessary, the peaks count matrix returned by the function `findpolypeaks` is passed to `rmshoulders` in order to remove or flatten the secondary or noisy peaks so-called shoulders in frequency distributions. The peaks count matrix returned by `findpolypeaks` and/or `rmshoulders` is passed to `.estimatek` for estimating k using different descriptive statistics. Finally, a set of estimates of k is returned to the function `findk`.

TABLE II. Functions of Kpeaks

Function	Description
.estimatek	Computes the estimates of k internally.
findk	Returns the estimates of number of clusters in a dataset.
genpolygon	Generates the classes to build a frequency distribution of features.
findpolypeaks	Finds and counts the peaks of frequency distribution of each feature.
rmshoulders	If any, removes the shoulder peaks in frequency distributions.
plotpolygon	Plots frequency polygons of the features.
4p5c	Synthetic dataset with 5 features and 4 clusters for testing purposes.

The package 'kpeaks' can be installed from CRAN and loaded into R working space with the following R commands.

```
> install.packages('kpeaks')
> library(kpeaks)
```

A. Synthetic Dataset

The functionalities of the package 'kpeaks' is demonstrated on synthetically created dataset. This dataset named *x5p4c* is included in the package and can be loaded into R workspace as follows.

```
> data(x5p4c)
```

The dataset `x5p4c` is a numeric dataset that consists of 400 rows and 5 columns. The structure and content of the first five rows of the dataset are shown below.

```
> str(x5p4c)
  'data.frame':   400 obs. of  5 variables:
   $ p1: num  7.21 11.05 10.28 8.31 9.18 ...
   $ p2: num  40.3 49.7 40.6 36.6 36.6 ...
   $ p3: num  101 101 104 103 110 ...
   $ p4: num  17.9 20.1 18.5 15.5 16.5 ...
   $ p5: num  19.2 19 19.1 19 19.1 ...

> head(x5p4c, 5)
          p1       p2       p3       p4       p5
1  7.206447 40.32824 101.0966 17.89194 19.18859
2 11.046624 49.72240 101.2886 20.08852 19.01149
3 10.284529 40.61714 104.1577 18.50747 19.06391
4  8.306704 36.58646 102.8601 15.53279 18.96799
5  9.176357 36.58312 109.6906 16.54572 19.07434
```

The features named as `p1`, `p2`, `p3`, `p4` and `p5` in the dataset have been created by using the package 'MixSim' [9] to form 4 clusters and frequency distributions with 1, 4, 3, 4 and 2 modes respectively, as seen in Fig. 2.

B. Generating Classes of Frequency Distribution

The function `genpolygon` of 'kpeaks' constructs the frequency distribution of a feature by using a selected binning rule, and returns the middle values and frequencies of classes for further processing. The usage syntax of the function is as follows.

Usage: `genpolygon(x, binrule, nbins, disp=FALSE)`

The arguments of the function are:

- `x` is a numeric vector containing the observations for a feature.
- `binrule` is the name of selected rule in order to compute the number of classes to build the frequency distribution. The available rules with `genpolygon` are listed in Table 3.
- `nbins` is an integer describing the number of classes. Except the rule `usr`, it is automatically computed by using the related formula of the selected binning rule shown in Table 3. If the `binrule` is `usr` it should be assigned by the user.
- `disp` is a logical flag whether the histogram and polygon of the distribution are displayed or not.

Most of the rules listed simply include only *n,* number of observations as the input argument in order to build frequency tables in statistics. On the other hand, the rules using the measures of variability and shape of data distributions may provide more optimal *k* values. For instance, Doane [10] extended the Sturges's rule [11] by adding the standardized skewness in order to overcome the problem with non-normal distributions needing more classes. In order to estimate optimal *k* values, Scott [12] added the standard deviation to his formula. Freedman and Diaconis [13] proposed to use the interquartile range (IQR) statistic which is insensitive to outliers when compared to standard deviation. In a study on unsupervised discretization methods, Cebeci and Yildiz [14] tested a binning rule formula based on the ten-base logarithm of *n* divided by 2π. The function `genpolygon` uses 'sturges' as the default rule.

TABLE III. ESTIMATION RULES OF NUMBER OF CLASSES

Name of rules	Formula	binrule
Square root [15]	$\lfloor n^{1/2} \rfloor$	sqr
Sturges [11] Huntsberger [16]	$\lceil 1 + log_2 n \rceil \cong$ $\lceil 1 + 3.332\ log_{10} n \rceil$	sturges huntsberger
Brooks-Carruthers [17]	$\lceil 5\ log_{10} n \rceil$	bc
Cencov [18]	$\lceil n^{1/3} \rceil$	cencov
Rice [19]	$\lceil 2\ n^{1/3} \rceil$	rice
Terrell-Scott [20]	$\lceil (2n)^{1/3} \rceil$	ts
Scott [12]	$\lceil R\ /\ 3.5\ \hat{\sigma}\ n^{-1/3} \rceil$	scott
Freedman-Diaconis [13]	$\lceil R\ /\ 2\ \ IQR\ n^{-1/3} \rceil$	fd
Doane [10]	$1 + log_2 n + log_2\left(1 + \frac{\lvert g_1 \rvert}{\sigma_{g_1}}\right)$; $\sigma_{g_1} = \left(\frac{6(n-2)}{(n+1)(n+3)}\right)^{1/2}$	doane
Cebeci [14]	$\lceil log_{10} n / 2\pi \rceil$	cebeci
-	a user specified integer	usr

In the following example, generation of frequency table of the feature `p2` is demonstrated. As it is seen from this example, the function `genpolygon` returns the frequencies (`freqs`), middle values (`mids`) and number of classes (`nbins`) as the result. These values are passed to the other functions of 'kpeaks' for further processing.

```
> hvals <- genpolygon(x5p4c$p2, binrule="sturges")
> print(hvals)
$freqs
[1]  1 55 44 56 44 54 45 60 41

$mids
[1]  25  35  45  55  65  75  85  95 105

$nbins
[1] 9
```

C. Plotting Peaks

The function `plotpolygon` of 'kpeaks' plots the histograms and frequency polygons with the following usage syntax.

```
plotpolygon(x, nbins, ptype, bcol="gray", pcol="blue")
```

In the following example, plotting of the scatterplots of feature pairs is demonstrated. In the diagonal of the Fig. 2, the histograms and frequency polygons of the features are displayed to examine the structure and shape of the features.

```
> pairs(x5p4c, diag.panel=plotpolygon,
+ upper.panel=NULL, cex.labels=1.5)
```

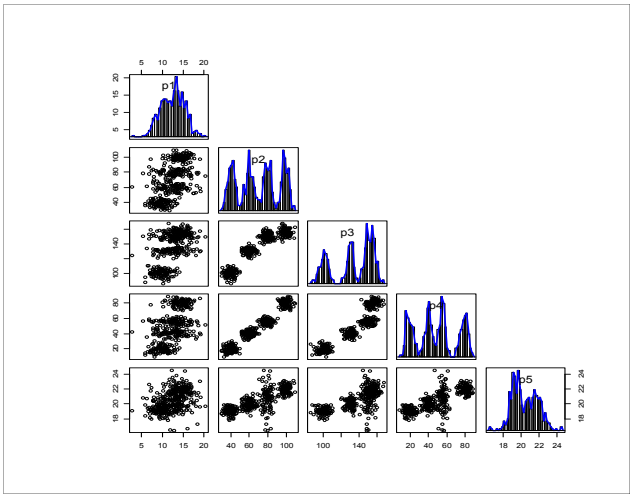

Fig. 2. Pairwise scatter plots of the features in *x5p4c* dataset

D. Finding and Counting Peaks

The function `findpolypeaks` in the package 'kpeaks' detects the peaks in a frequency polygon by using the frequencies and middles of the classes which are given as the input arguments as shown in the usage syntax as follows.

Usage: `findpolypeaks(xm, xc, tcmethod, tc)`

The arguments of the function are:

- `xm` is a numeric vector that contains the middle values of the classes of the frequency distribution.
- `xc` is an integer vector that contains the frequencies of the classes of the frequency distribution.
- `tcmethod` is a string represents the threshold method to discard the empty and the small bins whose frequencies are smaller than `tc`, the threshold frequency. The methods in Table 4 can be used to compute a threshold frequency value using the descriptive statistics of the frequencies in `xc`.
- `tc` an integer which is used as the threshold frequency value for discarding the empty and classes with low frequency in the frequency distribution. The default value of `tc` is 1 if the threshold option `usr` is selected. It means that a class whose frequency is less than or equal to 1 is discarded from the frequency polygon before peak counting.

In the next example, first, the frequency distribution is computed with `genpolygon` and then passed to `findpolypeaks` to detect the peaks in this distribution using `min` thresholding method. As it is seen in Fig.3, `findpolypeaks` detects 9 peaks for the examined frequency distribution.

```
> hvals <- genpolygon(x5p4c$p2, binrule="usr",
+   nbins=40)
> resfpp <- findpolypeaks(xm=hvals$mids,
+   xc=hvals$freqs, tcmethod = "min")
> print(resfpp)
   $pm
        pvalues pfreqs
   [1,]      41     19
   [2,]      53     10
   [3,]      59     23
   [4,]      77     18
   [5,]      83     19
   [6,]      91      3
   [7,]      97     23
   [8,]     103     16
   [9,]     109      2

   $np
[1] 9
> plotpolygon(x5p4c$p2, nbins=hvals$nbins, ptype="p")
```

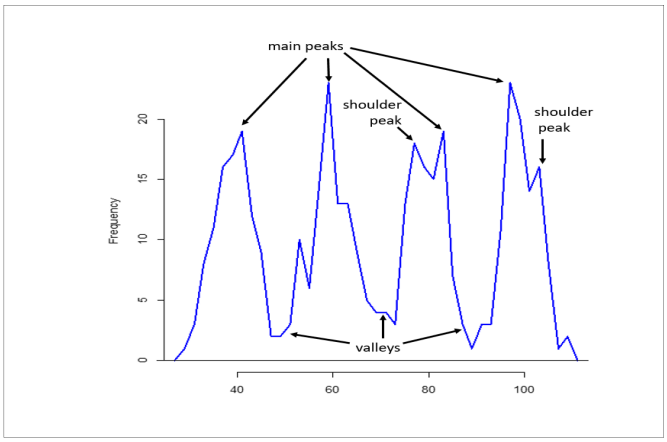

Fig. 3. Frequency plot for the feature p2 in x5p4c dataset

E. Removing Shoulder Peaks

As a geographical term, a "shoulder peak" or shortly "shoulder" is a secondary peak in a surrounding location close to the main peak of a mountain. Like the shoulder peaks of a mountain, a shoulder is a smaller peak that is quite close to the highest peak in a frequency distribution as seen in Fig 3. Shoulders may occur due to random noises or by selecting higher number of classes in class generation. Usually, it is desired to remove them from the peaks vector of a frequency distribution.

In the function `rmshoulder` of 'kpeaks', a peak is considered as a shoulder when its height is smaller than the height of its neighbor peak, and its distance to its neighbor is also smaller than a threshold distance value. In order to compute a threshold distance value, we propose to use seven options which are described below. The options `q1` and `iqr` can be applied to remove the minor shoulders that are very near to the main peaks while `q3` is recommended to eliminate the substantial shoulders in the processed frequency distribution. The remaining options may be more efficient for removing the moderate shoulders.

Usage: `rmshoulders(xm, xc, trmethod, tv)`

The arguments of the function are:

- `xm` is a numeric vector that contains the middle values of the classes of the frequency distribution.
- `xc` is an integer vector that contains the frequencies of the classes of the frequency distribution.
- `trmethod` is a string representing the type of shoulders removal option for computing a threshold value. The default method is `usr`. The alternatives are `sd`, `q1`, `iqr`, `avg` and `med`. These methods compute a threshold distance value using some statistics of the distances between the middle values of two successive peaks in the vector `xm`.
    - `sd` uses the standard deviation.
    - `q1` uses the first quartile (Q1).
    - `q3` uses the third quartile (Q3).
    - `iqr` uses the interquartile range (IQR).
    - `avg` uses the arithmetic mean.
    - `med` uses the median.
    - `usr` uses a user-specified number.

The function `rmshoulders` returns the following items as the output:

- `pm` is a data frame with two columns which are named `pvalues` and `pfreqs` containing the middle values and frequencies of the peaks which are determined by `findpolypeaks`, respectively.
- `np` is an integer representing the number of peaks in the frequency polygon.

Below, an illustrative example is given for demonstrating the shoulders removal process with `rmshoulders`. Comparing Fig. 3 and Fig. 4 it can be easily seen that `rmshoulders` successfully removes the shoulders by detecting the main peaks of frequency distribution.

```
> resrs <- rmshoulders(resfpp$pm[,1], resfpp$pm[,2],
+  trmethod = "q1")
> print(resrs)
   $pm
        pvalues pfreqs
   [1,]      41     19
   [2,]      59     23
   [3,]      83     19
   [4,]      97     23

   $np
   [1] 4
```

Compared to the output obtained in Section 3.B, only the main peaks exist in the output above. Since the function `rmshoulders` removes the secondary peaks or shoulders, the modes of frequency distributions can be more accurately counted to determine *k* with the function `.estimatek`.

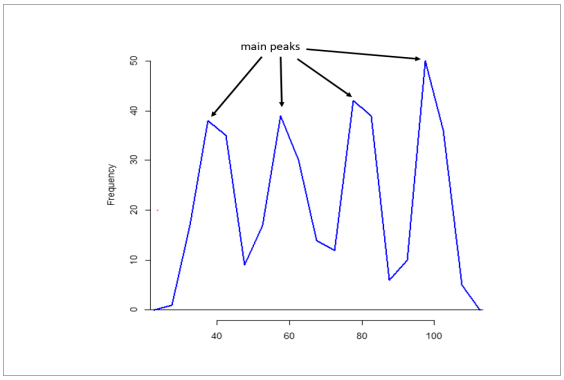

Fig. 4. Frequency polygon of the feature *p2* after shoulder removal

F. Estimating k

The function `findk` returns a list of *k* values which are proposed as the estimates of number of clusters in a given dataset. The estimation is based on various descriptive statistics of the peak counts in the frequency polygon of the features which have been previously listed in Table 1. The syntax of function `findk` is as follows.

Usage: `findk(x, binrule, nbins, tcmethod, tc, trmethod,tv, rms=FALSE, rcs=FALSE, tpc=1)`

The arguments of the function are:

- `x` is a numeric data frame or matrix.
- `binrule` is a string specifying the binning rule to compute the number of classes of a frequency polygon.
- `nbins` is an integer specifying the number of classes (bins). It is internally computed according to the selected binning rule except 'usr '.
- `tcmethod` is a string representing a threshold method to compute a threshold distance value to discard the small or empty bins of a frequency polygon. See all available options in findpolypeaks.
- `tc` is an integer for threshold frequency value assigned by `tcmethod`.
- `trmethod` is a string used to specify a removal method to discard the shoulders around the main peaks in a frequency polygon. The options for this argument is described in the section for `rmshoulders`.
- `tv` is a numeric threshold distance value assigned by `trmethod`. Its default value is 1 for cleaning the empty and low frequency classes.
- `rms` is a logical value whether the shoulders removal is applied or not. Default value is `FALSE` for processing without shoulder removal.
- `rcs` is a logical value whether the estimates of *k* are computed on the reduced counts set instead of the full set. Default value is `FALSE`, and set to `TRUE` in order to use the reduced counts set.
- `tpc` is an integer threshold value for creating the reduced set of the peak counts. Default value is 1.

978-1-5386-6878-8/18/$31.00 ©2018 IEEE

The function `findk` returns a list of the estimates of *k* consists of the following items which are computed from the peak counts of the features:

- `am` is the arithmetic mean of peak counts.
- `med` is the median of peak counts.
- `mod` is the mode of peak counts.
- `cr` is the center of the range of peak counts.
- `ciqr` is the center of the interquartile range (IQR) of peak counts.
- `mppc` is the overall mean of the pairwise means of peak counts.
- `mq3m` is the mean of the third quartile (Q3) and the maximum of peak counts.
- `mtl` is the mean of two largest values of peak counts.
- `avgk` is the proposed *k* as the mean of all the estimates.
- `modk` is the proposed *k* as the mode of all the estimates.
- `mtlk` is the proposed *k* as the mean of the two largest estimates.
- `dst` is a string representing the type of counts set which is used in computations.
- `pcounts` is an integer vector containing the peak counts of the features.

In the following example, `findk` is demonstrated to estimate the number of clusters in the dataset `x5p4c`. As a result, the function `findk` produces a list of all the computed estimates including proposed ones. The names of these components can be seen by using `attributes` function of R. Each name indicates the applied techniques listed in Table 1.

```
> estk <- findk(x5p4c, binrule="scott")
> attributes(estk)
$names
 [1] "am" "med" "mod" "mppc" "cr" "ciqr" "mq3m"
 [8] "mtl" "avgk" "modk" "mtlk" "dst" "pcounts"
```

In the following examples some of the estimates of *k* are listed. As it is seen in the output below, it is computed as 3 and 4 by the techniques of median technique (MED) and maximum two largest (MTL), respectively.

```
> print(estk$am)
 [1] 4
> print(estk$med)
 [1] 3
> print(estk$mppc)
 [1] 3
> print(estk$mq3m)
 [1] 4
> print(estk$mtl)
 [1] 4
```

According to the results listed above, the function `findk` proposes the estimates of *k* between 3 and 4. The users can select either one of these estimates of *k* or allow to the function to propose the optimal *k* by examining the individual results of techniques. For this purpose, the output components `avgk`, `modk` and `mtlk` can be used for final selection of *k*. In the following examples, the estimates of *k* are proposed as 4 by `avgk` and `mtlk`.

```
> print(estk$avgk)
 [1] 4
> print(estk$mtlk)
 [1] 4
```

According to the results above, the `findk` of 'kpeaks' successfuly determines the actual number of clusters in the test dataset `x5p4c`.

IV. Conclusions

In this paper, we introduced and illustrated the use of 'kpeaks' which is an R package enabling R users to estimate the *k* which is a mandatory input argument for partitioning cluster analysis. In the experiments on several real datasets such as Iris, Wine, Foresttype and Glass, downloaded from UCI [21] it was also found that 'kpeaks' provides accurate results in determining *k*. However the package targets unsupervised clustering as the first application domain, it also can be used as a useful tool in some other domains requiring to process the peaks of features in high dimensional datasets. For future versions of the package, a plan to enhance it with some additional functionalities such as benchmarking of computing time and improved visualization tools is under consideration.

Acknowledgment

This research was supported by the Scientific Research Projects Coordination Unit at the Çukurova University (Grant # FBA-2017-9730).

The package 'kpeaks' can be downloaded from CRAN at https://CRAN.R-project.org/package=kpeaks.

References

[1] G. Hamerly and C. Elkan, 'Learning the k in k-means', in *Advances in Neural Information Processing Systems*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. MIT Press, 2004, pp. 281–288.

[2] T. M. Kodinariya and P. R. Makwana, 'Review on determining number of Cluster in K-Means Clustering', *Int. J. Adv. Res. Comput. Sci. & Manag. Stud.*, vol. 1, no. 6, pp. 90–95, 2013.

[3] F. Kovács, C. Legány, and A. Babos, 'Cluster validity measurement techniques', in *Proc. 6th Int. Symp. of Hungarian Researchers on Computational Intelligence*, 2005.

[4] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, 'Internal versus external cluster validation indexes', *Int. J Comput. Commun.*, vol. 5, no. 1, pp. 27–34, 2011.

[5] A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. C. Tseng, 'Evaluation and comparison of gene clustering methods in microarray analysis', *Bioinformatics*, vol. 22, no. 19, pp. 2405–2412, 2006.

[6] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, 'On clustering validation techniques', *J. Intell. Inf. Syst.*, vol. 17, no. 2, pp. 107–145, 2001.

[7] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, 'NbClust Package for R', 2014.

[8] R Core Team, 'R: A Language and Environment for Statistical Computing'. Vienna, Austria, 2017.

[9] V. Melnykov, W.-C. Chen, and R. Maitra, 'MixSim: An R Package for Simulating Data to Study Performance of Clustering Algorithms', *J. Stat. Softw.*, vol. 51, no. 12, pp. 1–25, 2012.

[10] D. P. Doane, 'Aesthetic frequency classifications', *Am. Stat.*, vol. 30, no. 4, pp. 181–183, 1976.

[11] H. A. Sturges, 'The choice of a class interval', *J Am. Stat. Assoc.*, vol. 21, no. 153, pp. 65–66, 1926.

[12] D. W. Scott, 'On optimal and data-based histograms', *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[13] D. Freedman and P. Diaconis, 'On the histogram as a density estimator: L 2 theory', *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981.

[14] Z. Cebeci and F. Yildiz, 'Unsupervised discretization of continuous variables in a chicken egg quality traits dataset', *Turkish J Agric. Sci. Technol.*, vol. 5, no. 4, pp. 315–320, 2017.

[15] O. L. Davies and P. L. Goldsmith, *Statistical Methods in Research and Production*, 4th ed. Longman London, 1980.

[16] D. V Huntsberger, *Elements of statistical inference*. London: Prentice-Hall, 1962.

[17] C. E. P. Brooks and N. Carruthers, *Handbook of statistical methods in meteorology*. HM Stationery Office, 1953.

[18] N. Cencov, 'Evaluation of an Unknown Distribution Density from Observations', *Sov. Math.*, vol. 3, pp. 1559–1562, 1962.

[19] M. Lane, S. D, H. M, G. R, O. D, and H. Zimmer, 'Introduction to Statistics: A Multimedia Course of Study', 2016. [Online]. Available: http://onlinestatbook.com.

[20] G. R. Terrell and D. W. Scott, 'Oversmoothed nonparametric density estimates', *J. Am. Stat. Assoc.*, vol. 80, no. 389, pp. 209–214, 1985.

[21] M. Lichman, 'UCI: Machine Learning Repository' https://archive.ics.uci.edu/ml. 2013.

978-1-5386-6878-8/18/$31.00 ©2018 IEEE